Arquitectura de Computadores

2º Curso Grao Enx. Informática

Práctica 2

Programación Multinúcleo e extensións SIMD

Obxectivos: programar un algoritmo simple con vectores representados en punto flotante, utilizando diferentes graos de optimización (utilización de varios núcleos, utilización de extensións vectoriais simd, estratexias para reducir os fallos cache,..) e tomar medidas de rendemento combinando diferentes optimizacións e tamaños do problema.

Equipos: grupos de prácticas de ata dúas persoas.

Prazo de entrega: Mércores 6 de Maio 2020 ata as 14:00.

Valoración: 60% da nota de prácticas. Valoración por apartados (sobre 10): apartado i) e ii) → 1 punto, apartado iii) → 4.5 puntos facendo as opcións a) e b), 3 puntos facendo só unha das opcións, apartado iv) → 4.5 puntos.

Facer diferentes programas en C que realicen a computación do quaternion dp a partir dos vectores de quaternions a e b, indicado polo seguinte pseudocódigo:

Entradas:

```
vectores de quaternions a(N),b(N): N elementos de valor aleatorio coas
compoñentes dos quaternions de tipo double.
```

Saída:

```
dp: quaternions con componentes de tipo double.
```

Computación:

```
c(N): vector auxiliar de quaternions.
for i=1,N {
    c(i)=a(i)*b(i); // * indica multiplicación de quaternions.
}
dp=0; //inicialización do quaternion a cero (todas as compoñentes cero);
for i=1,N {
    dp=dp+c(i)*c(i) // * e +: multiplicación e suma sobre quaternions.
}
```

As diferentes versións son as seguintes:

i) Programa secuencial base (codificación en C do pseudocódigo anterior utilizando para cada quaternion catro variables tipo double)

ii) Programa secuencial optimizado (intentar modificar o código – o único que nos interesa é o valor final de dp - para obter algunha mellora no tempo de execución se é posible – cache, paralelismo a nivel de instrucción, unrolling, simplificación de operacións, etc)

iii) Programa secuencial optimizado utilizando extensións AVX para utilizar procesamento vectorial SIMD, seguindo a mellor (implementación con menor retardo) das seguintes estratexias: iii-a) vectorizar a multiplicación de dous quaternions, iii- b) vectorizar por iteracións do bucle

iv) Programa utilizando OpenMP para paralelizar a version secuencial optimizada (**non está permitido utilizar variables tipo "reduction"**), sen utilizar as extensións AVX, e variando o número de fíos (cores): 1, 2, 4, 6 e 8 (variar ata o máximo número de cores do voso PC).

Os códigos deberán compilarse do seguinte xeito:

i) Este código deberá compilarse sen optimizacións do compilador: `"gcc -O0 file_name.c"`, e con optimizacións incluída a autovectorización `"gcc -mavx -O3 file_name.c"`
ii) Este código deberá compilarse sen optimizacións do compilador: `"gcc -O0 file_name.c"`
iii) Estes códigos deberán compilarse sen optimizacións do compilador: `"gcc -mavx -O0 file_name.c"`. Engadir na cabeceira do código `"#include <immintrin.h>"` para que o compilador recoñeza as instruccións vectoriais.
iv) Estes códigos deberán compilarse sen optimizacións do compilador e co flag de OpenMP: `"gcc -O0 -fopenmp file_name.c"`

Facer varios experimentos utilizando os seguintes tamaños de vectores: $N=10^q$, con q=2, 4, 6 e 7. En todos os casos inicializar os vectores con valores aleatorios, e medir o número de ciclos da parte do programa na que se fai a computación indicada anteriormente. O almacenamento inicial dos datos debe ser igual en todas as versións. **Asegurarse que o resultado final en todas as versións é o mesmo, é dicir, para os mesmos valores iniciais o valor de dp é igual.** Cando se utilice OpenMP ou extensións AVX, debe incluírse na medida de tempo todo o que implique unha sobrecarga respecto do programa secuencial optimizado. Para cada caso, tomar 10 medidas, e seleccionar a mediana destes valores como valor final da medida.

As medidas de ciclos tomaranse nos vosos ordenadores. Deberedes indicar na memoria o tipo de procesador coas súas características principais. NOTA: Todos os procesadores desde o ano 2011 soportan as instrucións AVX (se algún alumno ou alumna non dispoñen dun ordenador posterior a ese ano, falar co profesor).

MEMORIA DA PRÁCTICA e CÓDIGOS FONTE: debe entregarse unha memoria (en PDF) cos resultados da práctica. A memoria debe conter unha introdución, que describa os obxectivos dos experimentos e as características do procesador utilizado. Para cada apartado debe conter o listado do código (non incluír as rutinas de medidas de ciclos), as gráficas obtidas, e unha interpretación das mesmas. Finalmente deben presentarse as conclusións finais a modo de resumo. ***Facer as representacións gráficas que se consideren oportunas para sacar conclusións dos resultados obtidos (é de especial interese a ganancia en velocidade da versión secuencial optimizada con respecto á versión inicial, a ganancia en velocidade dos códigos dos apartados iii), iv) e v) con respecto da versión secuencial optimizada, e finalmente a versión inicial compilada con -O3 con respecto a todas as demais).***

Os códigos fonte resultado dos apartados ii), iii) e iv) envíaranse o profesor dentro dun arquivo zip ou rar a través do Campus Virtual xunto coa memoria en PDF. O envío é obrigatorio (e dentro do prazo de entrega da memoria). **En caso de facer a práctica en grupo SÓ un membro do grupo debe enviar o código.**

CRITERIOS DE AVALIACIÓN: cumprimento das especificacións do enunciado da práctica, buscando sempre obter a mínima latencia (tempo de execución) do programa nas diferentes implementacións, rigurosidade no plantexamento e interpretación de resultados, exploración de alternativas, estudo autónomo e presentación de resultados (calidade da memoria). **Será considerado como moi negativo a copia directa de código (dende calquera fonte).**